uc3m | Universidad Carlos III de Madrid
e-Archivo

Metadata, citation
drid e-Archivo

This paper has been presented at:

Seventh International Workshop on Cloud Technologies and Energy Efficiency in Mobile Communication Networks (CLEEN 2019). How cloudy and green will mobile network and services be?
15 April 2019 - Marrakech, Morocco

https://5g-ppp.eu/cleen2019/

Profit Maximization by Forming Federations of Geo-Distributed MEC Platforms

Li-Hsing Yen, Chi-Han Chang, and Yi Chia Chen
Department of Computer Science, College of Computer Science, National Chiao Tung University, Hsinchu, Taiwan.
Email: lhyen@nctu.edu.tw, cross122911@gmail.com, yijia0112@hotmail.com

***Abstract*—Multi-access edge computing (MEC) as an emerging technology which provides cloud service in the edge of multi-radio access networks aims to reduce the service latency experienced by end devices. When individual MEC systems do not have adequate resource capacity to fulfill service requests, forming MEC federations for resource sharing could provide economic incentive to MEC operators. To this end, we need to maximize social welfare in each federation, which involves efficient federation structure generations, federation profit maximization by resource provisioning configuration, and fair profit distribution among participants. We model the problem as a coalition game with difference from prior work in the assumption of latency and locality constraints and also in the consideration of various service policies/demand preferences. Simulation results show that the proposed approach always increases profits. If local requests are served with local resource with priority, federation improves profits without sacrificing request acceptance rates.**

I. Introduction

Emerging Internet of Thing (IoT) applications demand massive computation and communication resource yet low response time, which is not viable with the current cloud environment. As a response, multi-access edge computing (MEC) [1] has been proposed, which brings cloud servers (namely MEC servers) to the proximity of IoT devices [2] while exploiting multi-radio access technology. MEC also facilitates other time-sensitive cloud services and applications like cloud-based AR/VR.

An MEC system comprises a set of inter-connected small-scale MEC servers that scatter over some region, and is operated and controlled by an MEC service provider (MSP). It is envisioned that in the near future, a couple of MEC systems each with limited resource capacity may have partially overlapping service coverage areas. When MEC user's resource requests come in bursts targeting at some hot-spot zone, the demand may exceed the capacity of a single MEC system. Meanwhile, other MEC systems may have residual capacities that can collectively meet the user's demands. Fig. 1 shows an example with four MSPs. Here MSP_1 and MSP_3 both have resource demands exceeding their capacities while MSP_2 and MSP_4 have residual capacities which are left idle. If these MSPs form a *federation* to enable resource sharing among them, all federation members can potentially have higher profits (with the consideration of resource costs and user's payments). The federation also benefits MEC users as it could lower request denial rate due to limited capacity.

In this work, we assume a broker who acts for all MSPs to form a *federations structure* [3]. A federation structure is

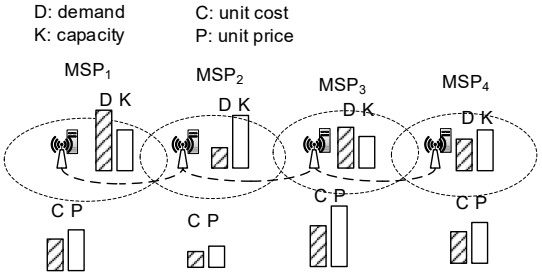


Fig. 1. Example of MEC Federation

a partition of all participants into disjoint subsets, each corresponding to a federation. Different federations yield different profits. If we define the *social welfare* of a federation structure to be the total profits in the structure, finding a federation structure that maximizes the social welfare is NP-complete [4]. Sandholm et al. [4] proposed an approach to this problem which involves three activities: federation structure generation, maximizing the profit of each federation, and allocating the profit of each federation to federation members.

Federation structure generation is non-trivial because the number of all possible federation structures is an exponential function of the number of participants [4]. A common heuristic to federation structure generation is *merge-and-split*, where the initial federation structure contains singleton federations. The approaches repeatedly merges two federations into more profitable one until no further merging is beneficial. It then attempts splitting every federation to seek possible profit improvement. If any federation is ever split, another round of merge-and-split then follows. The whole process stops when no further merging or splitting is possible.

The profit-maximization problem of each federation is domain-dependent. In most cases, it is a linear programming problem [5]. In our problem, this is to maximize the profit of a federation by *resource provisioning configuration* (RPC) in the federation subject to various constraints, policies, and preferences. It is not trivial because requests come with different demands and payments, and different MEC systems have different resource capacities and costs.

Finally, how federation profits are distributed to participants can affect the *stability* of federations. A federation is stable if no participant can be better off by deviating from the result (splitting from its federation to work alone or join another

federation). Generally speaking, computing a stable payoff configuration is intractable in most cases [4].

We propose an approach to federation structure formation in the framework of *coalition game*. Coalition game theory has been applied to grid computing [6], vehicular cloud [7], mobile cloud [8], traditional cloud [3], and fog computing [9]. Our approach differs from prior work in the following points.

- Assumption of latency and locality constraints imposed by requests. Without these constraints, any (cloud) server could serve any request provided that it has adequate resource. In contrast, an MEC server may not be able to serve a request due to long latency or lack of context information in an MEC environment.
- Assumption of independent service policies of service providers. An MSP may either serve local requests first (LRF) or seek maximal profit (MP) by serving more profitable requests from other MECs. On the other hand, MEC users may minimize latency by demanding local MEC service only (LSO) or maximize the ratio of granted requests by accepting any MEC service (AS). MEC users can also make a trade-off by requesting local service first (LSF) and accepting non-local service only when local MSP does not have enough resource.
- Consideration of heterogeneous service costs and prices. In traditional clouds, it is the service provider that determines selling price of resource. In our setting, it is MEC users that offer buying prices to MSPs.

The proposed scheme uses merge-and-split heuristic to form a federation structure that ensures stability. The maximal profit that can be earned by each federation is solved by an integer program solver. The profit of each federation is fairly distributed to federation members based on Banzhaf value [10]. We have conducted extensive simulations to study the performance of the proposed approach (in terms of profits and amount of allocated resource) with various settings. The results show that federations always increase profits. With LSF, we can increase total profits without any reduction on the amount of allocated resource in every MSP.

The rest of this paper is organized as follows: Sec. II formulates our problem and the next section presents the proposed approach in details. Simulation results and discussions are in Sec. IV. The last section concludes this work.

II. Problem Formulation

We assume MEC system providing Infrastructure as a Service (IaaS) with resource in the form of various sizes of virtual machines (Small, Large, XLarge, etc.) As in [5], we normalize the resources expressed in amounts of resources using the size of small instances as the reference. With this, Small, Large, and XLarge instances may demand 1, 2, and 4 units of resource, respectively (for example).

We assume a set of n MSPs $S = \{sp_1, sp_2, \ldots, sp_n\}$. Each MSP sp_i has a resource capacity C_i with unit cost c_i. We assume that all resource requests submitted by users of sp_i have been aggregated with total amount r_i and payment per unit p_i. These requests have latency constraint t_i. If sp_j serves requests from sp_i, the latency will be sp_j's computation latency l_j (depending on sp_j's computing power) plus communication latency $lc_{j,i}$. The offloading also incurs extra communication cost that could be estimated by the amount of resource provided by sp_j to sp_i times $b_{i,j}$, the unit cost of the communication link from sp_i to sp_j.

We assume that there is a broker for S which finds 1) a stable federation structure of S that maximizes social welfare, 2) a RPC for each federation that maximizes the total profit of the federation, and 3) a profit allocation scheme that fairly allocates the total profit of each federation to the federation members.

The social welfare of a federation structure $F = \{F_1, F_2, \ldots, F_m\}$ is defined as

$$u(F) = \sum_{F_i \in F} u(F_i), \tag{1}$$

where $u(F_i)$ is the maximum profit that F_i can earn. Let $\mathcal{F}(S)$ be set of all possible federation structures that can be formed on S. Given S, $F \in \mathcal{F}(S)$ is an *optimal federation structure* if it maximizes $u(F)$:

$$\max_{F \in \mathcal{F}(S)} u(F). \tag{2}$$

Let $q_{j,k}$ be the amount of resource provided by sp_j to sp_k. A RPC is a setting of all $q_{j,k}$'s for each sp_j and sp_k in a federation F_i. Apart from RPC, $u(F_i)$ also depends on the following parameters defined for every sp_j and sp_k in F_i:

- the unit profit when sp_j serves resource requests from users of sp_k, which is $p_k - c_j - b_{j,k}$, and
- whether the resource provided by sp_j to sp_k meets the associated latency constraint. We define an indication variable $f_{k,j}$ for this condition, where $f_{k,j} = 1$ if $l_j + l_{j,k} < t_k$ and $f_{k,j} = 0$ otherwise.

The value of $u(F_i)$ is the maximum profit that F_i can earn with any RPC:

$$u(F_i) = \max_{q_{j,k}} \sum_{sp_j \in F_i} \sum_{sp_k \in F_i} (p_k - c_j - b_{j,k}) \cdot q_{j,k} \cdot f_{k,j} \tag{3}$$

subject to *capacity constraint*

$$\sum_{sp_k \in F_i} q_{j,k} \leq C_j, \quad \forall sp_j \in F_i \tag{4}$$

and *demand constraint*

$$\sum_{sp_j \in F_i} q_{j,k} \leq r_k, \quad \forall sp_k \in F_i. \tag{5}$$

This is an integer programing problem. If the serving policy of some $sp_j \in F_i$ is LRF, we have additional constraint $\sum_{j \neq k} q_{j,k} \leq \max(C_k - r_k, 0)$ for it. If LSO is demanded in some $sp_k \in F_i$, we have $q_{j,k} = 0$ for all $j \neq k$ for it. If LSF is demanded instead, the constraint is $q_{k,k} = \min(C_k, r_k)$.

Profit distribution is to allocate $u(F_i)$ among all members of F_i. Let x_j be the profit allocated to each MSP $sp_j \in F_i$. An allocation $(x_j)_{sp_j \in F_i}$ is *feasible* if

$$u(F_i) = \sum_{sp_j \in F_i} x_j. \tag{6}$$

Let $(x_j)_{sp_j \in F_i}$ be a feasible allocation for F_i. If there exists another feasible allocation $(y_j)_{sp_j \in H}$ for some sub-federation $H \subset F_i$ such that $y_j \geq x_j$ for all $sp_j \in H$ and $y_k > x_k$ for some $sp_k \in H$, then H has a Pareto improvement on the allocation $(x_j)_{sp_j \in H}$. The existence of a Pareto improvement on the allocation of any subset $H \subset F_i$ implies instability of the allocation because all members in H could leave F_i to form a new federation without profit reduction and at least one member can receive a higher profit. In that case, we say that H *blocks* F_i. The goal of *stable* profit distribution is to find a feasible allocation for each federation F_i such that no $H \subset F_i$ can block F_i.

Algorithm 1 *merge-and-split*: Federation formation

$S \leftarrow \{\{sp_1\}, \{sp_2\}, \ldots, \{sp_n\}\}$
repeat
  $\mathcal{F} \leftarrow \{\{F_i, F_j\} | F_i, F_j \in S, F_i \vdash F_j \text{ or } F_j \vdash F_i\}$
  while $\mathcal{F} \neq \varnothing$ **do**
    repeat
      randomly select $\{F_i, F_j\}$ from $\mathcal{F}$
      $\mathcal{F} \leftarrow \mathcal{F} \setminus \{\{F_i, F_j\}\}$
    until *can_merge*(F_i, F_j) or $\mathcal{F} = \varnothing$
    if *can_merge*(F_i, F_j) **then**
      $S \leftarrow S \setminus \{F_i, F_j\}$
      $S \leftarrow S \cup \{\{F_i, F_j\}\}$
      $\mathcal{F} \leftarrow \{\{F_i, F_j\} | F_i, F_j \in S, F_i \vdash F_j \text{ or } F_j \vdash F_i\}$
    end if
  end while
  redo $\leftarrow$ *false*
  for all $H \in S$ such that $|H| > 1$ **do**
    for all partitions $\{F_i, F_j\}$ of H **do**
      if *can_split*(F_i, F_j) **then**
        $S \leftarrow S \setminus \{H\}$
        $S \leftarrow S \cup \{F_i, F_j\}$
        redo $\leftarrow$ *true*
        break
      end if
    end for
  end for
until *redo* = *false*
return S

III. Forming MEC Federations

A. Federation Structure Generation

As mentioned, we use merge-and-split as a heuristic to construct a federation structure. Refer to Algorithm 1. The algorithm forms the initial structure S that consists of singleton federations only, where each singleton federation is an MSP. Unlike in tradition clouds, where any two federations could be considered for possible merging, merging two federations F_i and F_j into one in MEC is beneficial only if some MSP in F_i is able to provide its resource to another MSP in F_j or vice versa subject to latency constraint. Recall that $f_{i,j} = 1$ if the request from sp_i can be served by sp_j without violating the latency constraint t_i. Based on f, we define *sharable* relation $\vdash$ on federations. For any two federations F_i and F_j, $F_i \vdash F_j$ iff $\exists sp_p \in F_i, \exists sp_q \in F_j, f_{p,q} = 1$. Therefore, merging F_i and F_j should be considered only if $F_i \vdash F_j$ or $F_j \vdash F_i$. We use $\mathcal{F}$ to keep the set of all possible pairs of federations in S for which merging should be considered.

The merging phase randomly picks up a pair of federations $\{F_i, F_j\}$ from $\mathcal{F}$ to see if F_i and F_j should be merged together. Function *can_merge*(F_i, F_j) returns the check result. On merging, S is updated by removing both $\{F_i\}$ and $\{F_j\}$ from it and adding the union of $\{F_i\}$ and $\{F_j\}$ into it. $\mathcal{F}$ is also updated accordingly.

For any two (possibly singleton) federations F_i and F_j such that $F_i \cap F_j = \varnothing$, a necessary condition for $H = F_i \cup F_j$ to be a stable federation is

$$u(H) \not< u(F_i) + u(F_j). \tag{7}$$

The reason is not hard to see. If (7) does not hold, either F_i or F_j blocks H for any feasible allocation for H. Even if (7) holds, whether H is stable also depends on the profit allocation for H. Let $x_k(F)$ denote the profit allocated to $sp_k \in F$. We define binary relation $\succeq$ on federations as

$$F \succeq F' \text{ iff } \forall sp_i \in F \cap F', x_i(F) \geq x_i(F') \tag{8}$$

and also relation $\equiv$

$$F \equiv F' \text{ iff } \forall sp_i \in F \cap F', x_i(F) = x_i(F'). \tag{9}$$

Finally, $F \succ F'$ if $F \succeq F'$ and $F \not\equiv F'$.

Some prior work allows merging F_i and F_j into H only if $H \succ F_i$ and $H \succeq F_j$ or $H \succ F_j$ and $H \succeq F_i$ [6], [7]. Some other work allows a merging only if the merging improves every member's profit [3]. We take the same merging rule as [3]. Refer to Algorithm 2.

Algorithm 2 Function: *can_merge*(F_i, F_j)

$H \leftarrow F_i \cup F_j$
for all $sp_k \in H$ **do**
  if $sp_k \in F_i$ and $x_k(H) \leq x_k(F_i)$ **then**
    return *false*
  else if $sp_k \in F_j$ and $x_k(H) \leq x_k(F_j)$ **then**
    return *false*
  end if
end for
return *true*

When there is no more federation pair in $\mathcal{F}$ to check, the algorithm proceeds to the splitting phase. It checks all possible partitions of every non-singleton federation H in S to see if H should be partitioned into two subsets. Whenever a partition occurs, the algorithm goes back to the merging phrase with the updated S.

Several conditions can be used for splitting up a federation H into two disjoint subsets F_i and F_j. The condition could be when the splitting improves at least one member's profit without decreasing any other's ($F_i \succ H$ and $F_j \succeq H$ or $F_j \succ H$ and $F_i \succeq H$) [7], when the splitting has a Pareto improvement on one subset ($F_i \succ H$ or $F_j \succ H$) [6], or when all members in one of the subsets have the same or higher profits after the splitting ($F_i \succeq H$ or $F_j \succeq H$) [3]. We take the same rule as [7]. So function *can_split*(F_i, F_j) returns *true* if $F_i \succ H$ and $F_j \succeq H$ or $F_j \succ H$ and $F_i \succeq H$, where $H = F_i \cup F_j$.

TABLE I
FOUR TYPICAL RESOURCE DEMAND/PROVISIONING CASES

		Service policy	
		MP	LRF
Request demand	LSO	**LSO**	
	LSF	-	**LSF**
	AS	**MP**	**LRF**

TABLE II
AN EXAMPLE OF FEDERATION FORMATION

MSP (sp_i)	sp_1	sp_2	sp_3	sp_4	sp_5
Resource Capacity (C_i)	90	80	140	100	250
Resource Demand (r_i)	120	100	120	180	100
Unit cost (c_i)	\$2	\$1	\$2	\$5	\$1
Unit price (p_i)	\$4	\$2	\$4	\$6	\$1

B. Resource Demand/Provisioning Cases

For a given federation, we need to find a RPC that maximizes the total profit. The RPC is subject to MSP's service policy and also user's demand preference. Recall that an MSP has two options for its service policy (LRF and MP) and each MEC user can demand either LSO, LSF, or AS. Table I lists four typical cases for resource demand and provisioning. We take the example shown in Table II to illustrate the difference of these four cases.

- Case 1 (LSO): When users only accept resource provided by local MSPs, it does not matter what the MSP's service policy is. There cannot be any federation at all and all MSPs work alone. For the example shown in Table II, the social welfare is \$600 and all unserved requests amount to 130 units.
- Case 2 (LSF): MEC users are willing to accept resource provided by remote MSPs only when local MSP does not have enough resource. On the other hand, MSP must provide enough resource to local requests before offering residual capacity to remote requests. This setting ensures that MEC users receive at least the same amount of resource as in LSO and the social welfare is at least the same as in LSO. In this case, sp_4 and sp_5 in Table II could form a federation. After meeting its local demands, sp_5 could offer 80 units of resource to sp_4 to earn extra profit $80 \times (\$6 - \$1) = \$400$. The social welfare is thus \$1000 and the amount of unserved requests reduces to 30 units.
- Case 3 (MP): Federation members collaborate to maximize total profit while MEC users accept resource provided by any member in the federation. In our example, sp_1, sp_2 and sp_3 can form a federation so sp_2 could provide all its resource (80 units) to meet sp_1's local request and earn a profit of $80 \times (\$4 - \$1) = \$240$. After that, sp_1 could provide another 40 units of resource to earn a profit of $40 \times (\$4 - \$2) = \$80$. sp_1 and sp_3 then provide their residual resource to the local request of sp_2 without extra profit. sp_4 and sp_5 also form a federation such that sp_5 provides 180 units of resource to sp_4's local request. That brings in $180 \times (\$6 - \$1) = \$900$. The social welfare is \$1460 and the amount of unserved requests is 60 units.
- Case 4 (LRF): MEC users are willing to accept resource offered by any MSP and the maximal amount of resource that MSP sp_i can provide to remote users is limited by $C_i - r_i$. The limitation does not imply that sp_i should offer $C_i - r_i$ units of resource to its local users. Other MSPs in the same federation with cheaper residual resource may serve its local requests. In our example, $\{sp_3, sp_4, sp_5\}$ forms a federation such that sp_3, sp_4, and sp_5 provide 20, 10, and 150 units of resource, respectively, to serve sp_4's local requests, yielding a total profit \$1080 in the federation. sp_1 and sp_2 both form singleton federations. The social welfare is \$1340 with 50 units of unserved requests.

TABLE III
SIMULATION PARAMETERS

Name	Description	Default Value
n	Number of MSPs	10
μ_k	Mean of C_i	1200
σ_k	Standard deviation of C_i	110
μ_r	Mean of r_i	1000
σ_r	Standard deviation of r_i	110
μ_c	Mean of c_i	500
σ_c	Standard deviation of c_i	110
μ_p	Mean of p_i	1000
σ_p	Standard deviation of p_i	110
p	Cooperation intensity	0.6

C. Profit Allocation

Shapley value [11] for payoff distribution in a coalition is based on marginal contributions of players considering all possible player joining orders. For a specific joining order, a player's marginal contribution is the change of the value of the coalition before and after the player joins the coalition. Shapley-value-based profit distribution is fair but incurs high computational cost.

Banzhaf value based on marginal contribution can also be used for payoff distribution that guarantees fairness. It needs only one (instead of all) player joining order to compute Banzhaf value and thus is more computationally efficient. For this reason, we adopt Banzhaf value for profit distribution. The Banzhaf value for $sp_j \in F_i$ is defined as

$$\beta_j(F_i) = \frac{1}{2^{m-1}} \sum_{H \subseteq F_i \setminus \{sp_j\}} [u(H \cup \{sp_j\}) - u(H)], \quad (10)$$

where $m = |F_i|$. The normalized Banzhaf value is defined as

$$B_j(F_i) = \frac{\beta_j(F_i)}{\sum_{sp_k \in F_i} \beta_k(F_i)}. \quad (11)$$

The profit of federation F_i allocated to sp_j is proportional to $B_j(F_i)$:

$$x_j(F_i) = u(F_i) B_j(F_i). \quad (12)$$

IV. Simulation Results

We have studied the performance of the proposed approach via simulations with various settings. The performance metrics include the social welfare and the amount of allocated resource in constructed federation structures. We used Gaussian distribution to generate the values of C_i, r_i, c_i, and p_i for each MSP sp_i. Table III lists the notations for the means and the standard deviations of these variables with their default values. We used Python library PuLP [12] as an integer programming solver for RPC in each federation.

The value of $f_{i,j}$ for every pair of MSPs (sp_i, sp_j) was randomly determined with $\Pr[f_{i,j} = 1] = p$. We refer to p as the *cooperation intensity* among MSPs (with default value 0.6). Fig. 2 shows how the social welfare and the total amount of allocated resource changed with increasing p. Because LSO allows no resource sharing, the performance with LSO is not affected by p. The performance with all other three cases improves as p increases. Among them, the highest social welfare is with MP while the largest amount of allocated resource is with LSF. The performance with LRF is between these two cases.

We then varied μ_r, the mean number of requested resource units, to study how the resource demand-to-supply ratio affects the performance. The results are shown in Fig. 3. When μ_r is less than $\mu_k = 1200$, the mean number of capacity units, the demands are lower than the supplies so both the social welfare and the amount of allocated resource increase linearly as μ_r increases. When $\mu_r \geq \mu_k$, the amount of allocated resource is limited by μ_k. Still, the social welfare could be improved by appropriate RPC as MF demonstrates in Fig. 3a.

We next fixed all parameters but μ_p, the mean unit price of resource, as a way to vary the price-to-cost ratio. From the results shown in Fig. 4, we can see that resource requests are generally fulfilled with LSO and LSF. On the other hand, more profits can be earned with MF and LRF. The extra profits come at the cost of low request acceptance rates. The cost is particularly significant when the price-to-cost ratio is low.

We also varied σ_k, the standard deviation of resource capacity, to see how the diversity of resource capacity affects the performance results. Fig. 5 shows the social welfare and the amount of allocated resource with different resource demand/provisioning cases. The social welfare generally decreases as σ_k increases. The result with MP is the highest among all, followed by LRF, LSF, and LSO (in that order). The amount of allocated resource also decreases as σ_k increases, but LSF outperforms the others. The performance of MP and LRF is pretty much similar, and much better than LSO.

Although the supply of resource generally exceeds the demand ($\mu_k > \mu_r$), a higher σ_k value implies a higher variation of the supply. The worst performance is with LSO because LSO allows no resource sharing. In contrast, MP maximizes total profits due to its flexibility in RPC, but it does not maximize the total amount of allocated resource. One reason is that it does not allocate resource from some MSP to another when that allocation yields zero profit (even though doing so increases the amount of allocated resource.) Consequently, LSF allocates the most amount of resource.

V. Conclusions

We have proposed an approach to profit maximization for federated MECs. This approach uses merge-and-split procedure for efficient federation structure generation, an integer programming solver for RPC in each federation that maximizes profit, and a fair profit allocation based on Banzhaf value. The merge-and-split procedure considers latency and locality constraints. The RPC considers four different cases for resource demand and provisioning, namely, LSO, LSF, MP, and LRF. We have conducted extensive simulations to study the impact of these constraints and rules on the performance of the federations. The results show that maximal profits can be earned with MP but sometimes at the cost of reduced amount of allocated resource. With LSF, the amount of allocated resource is not lower than the case of no federation yet the profits can potentially be improved.

Acknowledgements

This work was partially supported by the Ministry of Science and Technology, Taiwan, under grant numbers 106-2221-E-009-004 and by the H2020 collaborative Europe/Taiwan research project 5G-CORAL (grant number 761586).

References

[1] T. Taleb, K. Samdanis, B. Mada, H. Flinck, S. Dutta, and D. Sabella, "On multi-access edge computing: a survey of the emerging 5G network edge cloud architecture and orchestration," *IEEE Communications Surveys & Tutorials*, vol. 19, no. 3, pp. 1657–1681, 2017.

[2] J. Pan and J. McElhannon, "Future edge cloud and edge computing for Internet of Things applications," *IEEE Internet of Things Journal*, vol. 5, no. 1, pp. 439–449, Feb. 2018.

[3] L. Mashayekhy, M. M. Nejad, and D. Grosu, "Cloud federations in the sky: formation game and mechanism," *IEEE Trans. on Cloud Computing*, vol. 3, no. 1, pp. 14–27, Jan.-Mar. 2015.

[4] T. Sandholm, K. Larson, M. Andersson, O. Shehory, and F. Tohmé, "Coalition structure generation with worst case guarantees," *Artificial Intelligence*, vol. 111, no. 1-2, pp. 209–238, Jul. 1999.

[5] M. Hadji and D. Zeghlache, "Mathematical programming approach for revenue maximization in cloud federations," *IEEE Trans. on Cloud Computing*, vol. 5, no. 1, pp. 99–111, 2017.

[6] L. Mashayekhy and D. Grosu, "A merge-and-split mechanism for dynamic virtual organization formation in grids," *IEEE Trans. on Parallel And Distributed Systems*, vol. 25, no. 3, pp. 540–549, Mar. 2014.

[7] R. Yu, X. Huang, J. Kang, J. Ding, S. Maharjan, S. Gjessing, and Y. Zhang, "Cooperative resource management in cloud-enabled vehicular networks," *IEEE Trans. on Industrial Electronics*, vol. 62, no. 12, pp. 7938–7951, Dec. 2015.

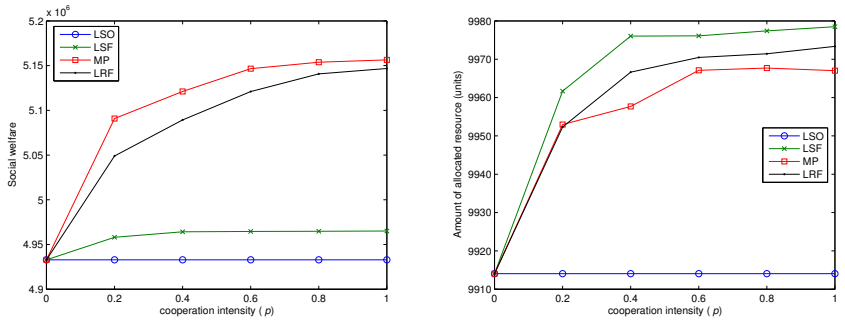
[8] R. Yu, J. Ding, S. Maharjan, S. Gjessing, Y. Zhang, and D. H. K. Tsang, "Decentralized and optimal resource cooperation in geo-distributed mobile cloud computing," *IEEE Trans. on Emerging Topics in Computing*, vol. 6, no. 1, pp. 72–84, 2018.

[9] C. Anglano, M. Canonico, P. Castagno, M. Guazzone, and M. Sereno, "A game-theoretic approach to coalition formation in fog provider federations," in *Proc. 3rd IEEE Int'l Conf. on Fog and Mobile Edge Computing*, Barcelona, Spain, Apr. 2018, pp. 123–130.

[10] G. Owen, "Multilinear extensions and the Banzhaf value," *Naval Research Logistics*, vol. 22, no. 4, pp. 741–750, Dec. 1975.

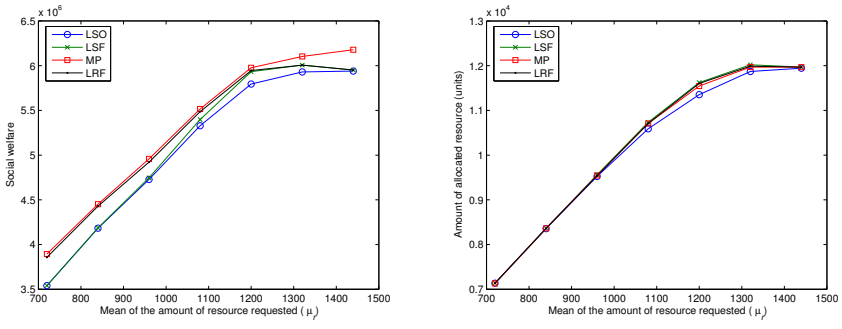
[11] L. S. Shapley, "A value for n-person games," in *Contributions to the Theory of Games II, Annals of Mathematics Studies*. Princeton University Press, 1953, vol. 28, pp. 307–317.

[12] "Optimization with PuLP." [Online]. Available: https://pythonhosted.org/PuLP/



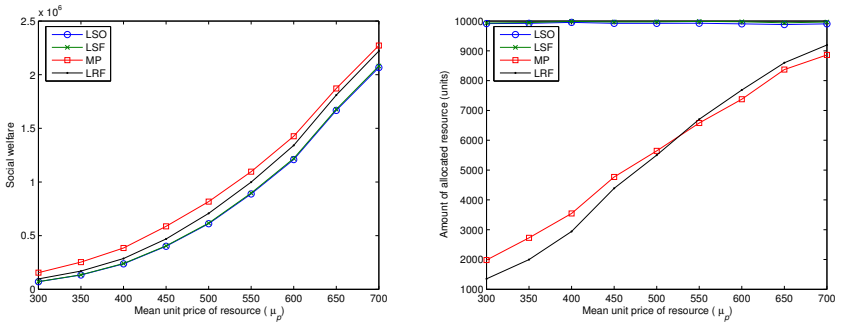
(a) (b)

Fig. 2. (a) Social welfare and (b) amount of allocated resource with respect to cooperation intensity (p)



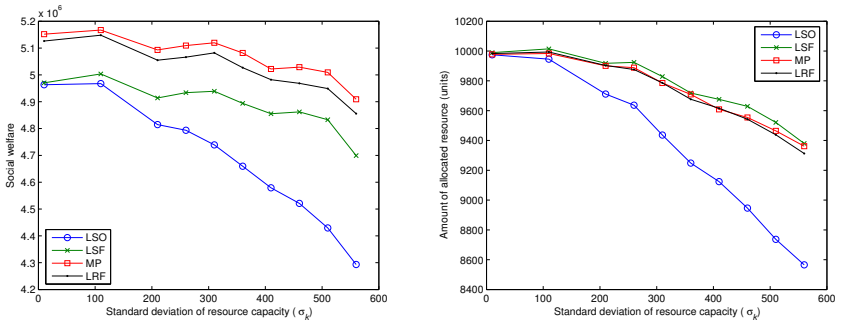
(a) (b)

Fig. 3. (a) Social welfare and (b) amount of allocated resource with respect to the mean units of resource request (μ_r)



(a) (b)

Fig. 4. (a) Social welfare and (b) amount of allocated resource with respect to the mean unit price of resource (μ_p)



(a) (b)

Fig. 5. (a) Social welfare and (b) amount of allocated resource with respect to the standard deviation of resource capacity (σ_k)